Input-Output and Hybrid LCA (Subject Editor: Sangwon Suh)

Efficient Algorithms for Life Cycle Assessment, Input-Output Analysis, and Monte-Carlo Analysis

Glen P. Peters

Industrial Ecology Programme, Norwegian University of Science and Technology (NTNU), Realfagbygget E1, 7491 Trondheim, Norway (glen.peters@ntnu.no)

DOI: http://dx.doi.org/10.1065/lca2006.06.254

Please cite this paper as: Peter GP (2007): Efficient Algorithms for Life Cycle Assessment, Input-Output Analysis, and Monte-Carlo Analysis. Int J LCA 12 (6) 373–380

Abstract

Goal, Scope, and Background. As Life Cycle Assessment (LCA) and Input-Output Analysis (IOA) systems increase in size, computation times and memory usage can increase rapidly. The use of efficient methods of solution allows the use of a wide range of analysis techniques. Some techniques, such as Monte-Carlo Analysis, may be limited if computational times are too slow.

Discussion of Methods. In this article, I describe algorithms that substantially reduce computation times and memory usage for solving LCA and IOA systems and performing Monte-Carlo analysis. The algorithms are based on well-established iterative methods of solving linear systems and exploit the power series expansion of the Leontief inverse. The algorithms are further enhanced by using sparse matrix algebra.

Results and Discussion. The algorithms presented in this article reduce computational time and memory usage by orders of magnitude, while still retaining a high degree of accuracy. For a 3225×3225 LCA system, the algorithm reduced computation time from 70s to 0.06s while retaining an accuracy of 10^{-3}%. Storage was reduced from 166 megabytes to 1.8 megabytes. The algorithm was used to perform a Monte-Carlo analysis on the same system with 1,000 samples in 90s. I also discuss various issues of power series convergence for general LCA and IOA systems and show that convergence will generally hold due to the mathematical structure of LCA and IOA systems.

Conclusions. By exploiting the mathematical structure of LCA and IOA iterative techniques substantially reduced the computational times required for solving LCA and IOA systems and for performing Monte-Carlo simulations. This allows more widespread implementation analysis techniques, such as Monte-Carlo analysis, in LCA and IOA.

Recommendations and Perspectives. It is suggested that algorithms, such as the ones described in this article, should be implemented in LCA packages. Various checks can be used to verify that computational errors are kept to a minimum.

Keywords: Algorithms; input-output analysis (IOA); matrix inverse; Monte-Carlo; numerical approaches; power series expansion; sensitivity analysis

Introduction

Life Cycle Assessment (LCA) and Input-Output Analysis (IOA) are becoming two prominent methodologies for analyzing the environmental impacts of consumer activities and production processes. LCA has traditionally focussed on detailed studies of particular products and services (Baumann and Tillman, 2004); such as paper cups (Hocking 1991). In contrast, IOA has traditionally focussed on more aggregated products and economic activities; such as household consumption (Hertwich 2005). Unfortunately LCA often suffers from ill-defined system boundaries (Lave et al. 1995) and consequently IOA is often used to increase the systems boundaries of LCA studies (Suh et al. 2004). Increasingly, LCA studies are either supplemented with IO data (e.g. Suh 2004) or performed almost entirely using IO data (Hendrickson et al. 1998, Joshi 1999).

With the increased use of LCA and IOA, there is also an increase in the size of the underlying databases. Currently, the biggest IO databases contained about 500 industry sectors (Bureau of Economic Analysis 2005) and some commercial LCA databases have data on around 2,500 processes (Frischknecht and Jungblush 2004). Given this, it is not uncommon for hybrid LCA systems to require the solution of equations with 2,000 to 3,000 variables. Even with modern computers, computational times for systems of this size can limit the applicability of some analysis techniques. For instance, detailed studies of parameter uncertainty using Monte-Carlo analysis may require the repeated solution of a system thousands of times (Huijbregts et al. 2003). For the wide spread use of numerical approaches, such as Monte-Carlo analysis, it is essential that computational resources are kept to a minimum.

Currently, there are two broad techniques for solving LCA systems (Heijungs and Suh 2002); the sequential approach and the matrix approach. The sequential approach starts with one central process and linearly scales the required products and processes; this is repeated further upstream. The matrix approach converts the LCA into a system of linear equations which are solved using standard linear algebra (Heijungs 1994, Heijungs and Suh 2002). IOA systems are only solved using matrix approaches (Miller and Blair 1985). Consequently, the use of the matrix approach allows the similarities of LCA and IOA to be exploited in hybrid LCA studies (Joshi 1999, Nakamura and Kondo 2002, Suh 2004, Peters and Hertwich 2006, Suh 2006). Further, by expressing LCA in the same matrix form as IOA allows easy application of the many established tools developed for IOA (e.g. Miller and Blair 1985, Lenzen 2003). For these reasons the remainder of this article will focus only on the matrix approach for LCA.

Generally, solving an LCA system using the matrix approach requires the solution of a system of linear equations (Heijungs

1994, Heijungs and Suh 2002). Direct methods of solving a system of linear equations include matrix inversion and Gaussian elimination (Nicholson 1990). However, it is well established that these direct methods are computationally inefficient (c.f. Watkins 2002). Slow computation times have made some LCA interpretation methods less viable. For instance, Heijungs et al. (2005) limit the number of samples used for Monte-Carlo analysis due to computational constraints as opposed to accuracy considerations. It is well established that for large systems of linear equations iterative methods are considerably faster then direct methods of solution (Watkins 2002).

This article discusses algorithms for the efficient solution of LCA and IOA systems and demonstrates the methods using Monte-Carlo analysis. Instead of using direct methods to solve systems of linear equations, I apply well established iterative methods which are computationally superior without compromising accuracy. Further, I discuss how the structure of IOA and LCA equations ensure the convergence of iterative methods. Most of the analysis draws upon well established results in the numerical analysis of linear systems (e.g. Berger and Saibel 1957, Stewart 1973, Seneta 1973, Hackbusch 1994, Datta 1995, Saad 1996, Varga 2000, Watkins 2002). I demonstrate the algorithms using Monte-Carlo analysis.

1 Solution of Linear Systems

For a given LCA, a system of linear equations can be written in the same form as in IOA[1]

$$x = Ax + y \tag{1}$$

where each element of the vector x is the output of the various production activities, each element of the vector y is the demand of products on the LCA (the functional unit), and A is the normalized technology matrix. In the normalized form the i-th column of A represents the inputs of products required to produce *one unit* of the i-th product. I assume that there is the same number of production activities denoted by n.[2] The normalized form in (1) is subtly different to the un-normalized equations used by some authors (e.g. Heijungs and Suh 2002). As I will show, to use iterative methods of solution as described below it is essential to have a normalized system for convergence to hold in an arbitrary system[3]. In **Appendix A** I describe how to construct a normalized system of linear equations from an un-normalized system.

Analytically the solution of (1) can be obtained by matrix inversion,

$$x = (I - A)^{-1} y = Ly \tag{2}$$

In the Leontief inverse, $L = (I\text{-}A)^{-1}$, the column i represents the required products to produce a unit of product i. The Leontief inverse has received relatively little attention in LCA, despite its importance in IOA (Miller and Blair 1985). If the environmental impacts per unit output, F, are known for each activity, then the total environmental impacts for the functional unit, y, are

$$f = F\,(I - A)^{-1}\, y \tag{3}$$

where the rows of F represent different environmental stressors and the columns represent the different production processes. Consequently, f is a column vector with each element representing a different environmental stressor.

While calculating the inverse, as in (2), or using Gaussian elimination are effective direct means of solving (1) they are not computationally efficient (Watkins 2002). Rather, iterative methods are often used to solve (1). Interestingly, expressing (2) in the form (1) is the starting point for many iterative numerical methods[4] of solving equations of the form (2) (c.f. Berger and Saibel 1957, Stewart 1973, Seneta 1973, Hackbusch 1994, Datta 1995, Saad 1996, Varga 2000, Watkins 2002). These methods are generally of the form

$$\begin{aligned} x_0 &= y \\ x_{i+1} &= Ax_i \quad i \geq 0 \end{aligned} \tag{4}$$

$$x = \sum_{i=0}^{\infty} x_i \tag{5}$$

Solving this iterative equation for $i \to \infty$ leads to the solution

$$x = y + Ay + A^2 y + A^3 y + \ldots \tag{6}$$

This power series expansion is commonly used in IOA as a way of interpreting the Leontief inverse (Miller and Blair 1985, United Nations 1999). The i-th term in the expansion represents the required products in the i-th production layer. In the zero-th production layer, the output y is from the manufacturing site or sites. To produce y an input of Ay is required into the first tier production layer. To produce Ay an input of $A(Ay) = A^2y$ is required into the second tier production layer, and so on through the infinite expansion.

In terms of IOA, the power series expansion

$$(I - A)^{-1} = I + A + A^2 + A^3 + \ldots \tag{7}$$

was also derived independently by Waugh (1950) and it was further shown that the series converges if A is a non-negative matrix and if the row sum of A is less than one,

$$\sum_{i=1}^{n} |\, A_{ij} \,| < 1 \quad \text{for all } j \tag{8}$$

[1] A detailed description on how to construct an LCA system as a system of linear equations can be found in Heijungs and Suh (2002); however, they construct an un-normalized system in the form $\tilde{A}s = \tilde{y}$. This is discussed further in Appendix A.

[2] For LCA, Heijungs and Suh (2002) describe how to deal with systems with different numbers of processes and products. In IOA the make and use system describes how to deal with secondary production (Miller and Blair 1985, United Nations 1993).

[3] Heijungs and Suh (2002), pp. 108–109, give an illustrative example of this.

[4] Many different iterative methods are defined based on how the iterative 'A' matrix is constructed from the starting point $x = Ly$. For simplicity I will assume that a system in the form (2) is rewritten as (1). The references describe the many methods of constructing an 'A' matrix.

This convergence condition holds for IOA systems in monetary units, but is not directly useful in mixed unit IOA or LCA since the matrices generally do not satisfy these conditions. A more useful result from linear algebra is that (7), hence (6), converge when the spectral radius

$$\rho(A) = \{\max(|\lambda|) \text{ for all eigenvalues, } \lambda, \text{ of } A\} \quad (9)$$

is less than one[5] (e.g. Seneta 1973, Datta 1995, Varga 2000). Further, this result does not require A to be non-negative. The spectral radius can also be used as an alternative to the Hawkins-Simon conditions (Hawkins and Simon 1949) for testing the existence and uniqueness of positive solutions to (1) (Wood and O'Neill 2002).

The iterative scheme (4) and power series expansion (6) offer an alternative way to solve (1). For most LCA systems it is generally straightforward to verify convergence (see below), however, of more interest is whether (4) is a computationally superior way of solving (1), particularly for larger systems.

Computational methods for solving linear systems of equations have been studied in considerable detail in the past. For an arbitrary system of equations in the form (1) it is generally found that direct calculation of the inverse takes $\frac{8}{3}n^3 + O(n^2)$ computational flops to solve[6], Gaussian elimination takes $\frac{2}{3}n^3 + O(n^2)$ flops, and iterative methods usually take Cn flops for some constant C which depends on the number of iterations required for convergence[7] (Hackbusch 1994, Saad 1996, Watkins 2002). The constant C can be large and so iterative methods are generally better for large n. If the linear system of equations is sparse[8] then these computational times are found to be significantly reduced (Hackbusch 1994, Saad 1996, Watkins 2002), although the computational times depend on the degree of sparsity. Overall, it is generally found that on most large problems the savings in computation times and storage achieved by using iterative methods 'are truly spectacular' (Watkins 2002).

2 Algorithms

In this section I describe simple algorithms that apply the iterative scheme in (4) and then compare the computational times to direct methods of solution. I also discuss the issues of convergence and accuracy.

Using a simple recursive procedure it is possible to implement (4) as in Algorithm 1. To reduce computational time and memory usage, the properties of sparse matrices are exploited[9] (Hackbusch 1994, Saad 1996). Algorithm 1 takes inputs of A, y, the desired accuracy of the solution (*tolerance*), and the maximum number of iterations to perform (*max_iterations*). The first four lines of the algorithm initialize the variables; *test* is given a value to ensure that at least one iteration occurs. The algorithm then recursively constructs the output until the desired accuracy is reached or the maximum iterations are exceeded. The index variable i is the current tier of the expansion (6), x_i holds the value of the current tier which is constructed recursively from the previous tier, see (4) and (6), x contains the total output, and the accuracy is measured relative to the previous iteration (*test*). It is also possible to calculate the Leontief inverse by replacing y with the identify matrix in Algorithm 1.

Typically the objective of an LCA is to calculate the environmental impact for a given functional unit, y. This allows minor modification of Algorithm 1 to allow direct calculation of environmental impacts, Algorithm 2. The structure of the algorithm is the same as Algorithm 1. The main difference is that an extra step is used to calculate the environmental impact. In terms of computational time, Algorithm 2 is superior due to the faster computation of *test*, and further, it can be used to give both the output and the environmental impacts.

Algorithm 1: CALCULATE_OUTPUT(A, y, *tolerance*, *max_iterations*)

$i \leftarrow 0$
$x_i \leftarrow y$
$x \leftarrow x_i$
$test \leftarrow tolerance$
while *(test > tolerance)* **and** *(i < max_iterations)*
  $i \leftarrow i+1$
  $x_i \leftarrow A * x_{i-1}$
  $x \leftarrow x + x_i$
  $test \leftarrow \frac{\|x_{i-1} - x_i\|}{\|x\|}$
end
return x

Algorithm 2: CALCULATE_IMPACT(F, A, y, *tolerance*, *max_iterations*)

$I \leftarrow 0$
$x_i \leftarrow y$
$f_i \leftarrow F * x$
$test \leftarrow tolerance$
while *(test > tolerance)* **and** *(i < max_iterations)*
  $i \leftarrow i+1$
  $x_i \leftarrow A * x_{i-1}$
  $x \leftarrow x + x_i$
  $f_i \leftarrow F * x$
  $test \leftarrow \frac{|f_{i-1} - f_i|}{|f|}$
end
return f_i

[5] In words, the spectral radius is the largest absolute value of the eigenvalues of A. In physical terms, if all the eigenvalues of A lie inside a disc of radius one then the power series converges. This is analogous to $\frac{1}{1-x}$ converging for $|x| < 1$. A number λ is called an eigenvalue of A if it satisfies $A\nu = \lambda\nu$ for some non-zero vector ν. Eigenvalues and eigenvectors have numerous applications in linear algebra (c.f. Nicholson 1990).

[6] The convention is usually to assume that the computational work is the same for additions, subtractions, multiplications, and divisions. The big-O notation, $O(n^2)$, essentially states that, in the limit of large n, the computational time of the algorithm varies proportional to n^2.

[7] It is also possible to determine the required iterations in advance (Seneta 1973, Datta 1995, Varga 2000); although a more efficient method is to add terms until an acceptable error or number of iterations is reached (see later).

[8] A sparse matrix has a high proportion of zero elements compared to non-zero elements.

[9] Many computational packages, for instance Matlab, have optimized routines for sparse matrix calculations.

2.1 Computational times

To give an indication of the computational times required for direct methods of solution compared to indirect iterative methods of solution, several calculations were performed using Matlab Version 7 operating under Microsoft Windows XP (**Table 1**). Of course, depending on the machine, the operating system, the operating state of the machine, and the code used for the calculations, these times will vary. The *A* matrix for this calculation was 3225×3225 and 1.4% of the matrix elements were non-zero. The data set is a hybrid LCA system based on ecoinvent (Strømman et al. 2006). The functional unit, *y*, has one non-zero element. The table clearly illustrates that the power series expansion gives more than sufficient accuracy and computational times three orders of magnitude faster than conventional techniques. The *error* in Table 1 compares the direct method of solution with the iterative method of solution. For a matrix system of this size it is not feasible to analytically calculate the solution and so the direct method is also a numerical approximation (see below for more details). The table shows that the *tolerance* gives a good indication of the *error* of the iterative method compared to the direct method.

In terms of memory usage, the use of sparse matrix algebra saves considerable space. For the system used here with 64-bit (8 byte) real numbers requires 3225×3225×8 = 83 megabytes of storage. The same storage would be required for the identity matrix. If the matrices are stored as sparse matrices then *A* requires 1.8 megabytes and the identity matrix requires 51 kilobytes. These savings may also lead to reduced computational time since system resources are available for other computational activities.

2.2 Convergence

In general, the convergence of iterative schemes can be verified using a variety of techniques; however the structure of LCA and IOA systems makes convergence easier to verify. With some conditions it is possible to prove that for an arbitrary LCA or IOA system $\rho(A) < 1$ and convergence holds. For the case of IOA monetary units it has been shown that converge always holds (Waugh 1950). In general, LCA and IOA systems are in mixed units so the row and column sums cannot be used to ensure convergence as they may be greater than one. In the case of mixed units convergence follows if *A* is a positive matrix. Given the structure of (1), it follows that *I* and *A* are a regular splitting of the system (e.g. Saad 1996, pp 107) and thus if $L > 0$ then it follows that $\rho(A) < 1$ (Saad 1996, also see O'Neil and Wood 1999, Wood and O'Neil 2002).

In some situations *A* may have non-negative coefficients. This usually arises due to assumptions on waste-streams and secondary production in both LCA (Heijungs and Suh 2002) and in IOA (Miller and Blair 1985). This generally occurs in very few elements and experience has shown this is generally not a problem in IOA (United Nations 1999). From a theoretical point of view, it may be difficult to guarantee convergence if elements are arbitrarily non-negative. Although, from a practical point of view it would be expected that convergence holds. If convergence did not hold, then the interpretation of the power series expansion (6) would not hold which would suggest that the *A* matrix was not constructed correctly. That is, the divergence of A^i as $i \to \infty$ implies that in the *i*-th production layer requires an infinite input of products. Thus it is likely that $\rho(A) < 1$ holds for all realistic LCA and IOA systems. From this practical point of view, if convergence did not hold then the validity of *A* would be questionable.

Given possible doubts about the convergence of *A* it is possible to use a variety of methods to test convergence (Saad 1996). For instance, it could be numerically verified that $\rho(A) < 1$, the matrix norm is less than one, or the matrix is diagonally dominant. Again, direct calculation is computationally slow and iterative methods are usually used, such as the power method[10] (the algorithms can be found in Datta 1995, Wood and O'Neil 2004). Using the power method, together with the properties of sparse matrices, reduces computation times of the spectral radius by orders of magnitude whilst still retaining accuracy (**Table 2**). Thus, computationally it is straightforward to verify if $\rho(A) < 1$.[11]

[10] Not to be confused with the power series expansion.

[11] Although, the pragmatic may skip this step and if (6) does not converge then it probably means that r.

Table 1: The computational times for CALCULATE_IMPACT. The asterisk, *, indicates the number of iterations to get an error of 10^{-3} percent. The direct calculations were performed using the Matlab functions inv which performs a matrix inverse and \ which uses Gaussian elimination. The error is relative to the direct method and the tolerance is from Algorithm 2 and compares the last two iterations

Tiers	Time	Time	Error (%)	Tolerance (%)
	full (s)	sparse (s)		
Direct, inv	70	40	–	–
Direct, \	14	4.3	–	–
10	0.6	0.03	2.3	1.4
20	1.2	0.05	3.3×10^{-2}	1.8×10^{-2}
27*	1.5	0.06	2.0×10^{-3}	9.7×10^{-4}
50	2.9	0.10	2.7×10^{-7}	1.3×10^{-7}
100	5.0	0.15	3.7×10^{-9}	8.6×10^{-11}

Table 2: The computational times for determining the spectral radius. The asterisk, *, indicates the number of iterations to get an error of 10^{-3} percent. The Direct calculation was performed using the Matlab call max(abs(eig(A))). The error is compared to the direct solution and the tolerance is the comparison of the last two iterations. The power method was used for the iterative solutions (e.g. Datta 1995, Wood and O'Neil 2004)

Iterations	Time	Time	Error (%)	Tolerance (%)
	non–sparse (s)	sparse (s)		
Direct	100	–	–	–
10	0.6	0.03	40	11
20	1.3	0.06	14	11
50	3.2	0.15	8.4×10^{-4}	4.1×10^{-3}
51*	3.2	0.17	3.7×10^{-4}	4.7×10^{-4}
100	6.5	0.31	9.7×10^{-8}	2.5×10^{-7}

2.3 Accuracy

Another issue of importance is the accuracy of the method. I have designed the algorithms to continue until a desired accuracy or maximum number of iterations is reached. If the matrix is *ill-conditioned* then both the direct and iterative solution *may be* sensitive to small changes in the functional unit. The condition number of a matrix is given by

$$\kappa(A) = \|A\|\|A\|^{-1} \tag{10}$$

where $\|A\|$ is a matrix norm and it follows that $\kappa(A) \geq 1$. A matrix is defined as ill-conditioned if $\kappa(A) >> 1$ (Watkins 2002). However, an ill-conditioned matrix is not always sensitive to small changes (Watkins 2002), particularly for sparse systems where the matrix columns consists mainly of zeros and are therefore nearly linearly-dependent.

The condition number for the matrix used in this article is of the order 10^{23} which suggests that the problem is highly ill-conditioned. However, by back-substituting x back into (1) gives an error of 10^{-2}% showing that the result is in fact accurate. In addition, the results presented in Table 1 show that the system is converging to the same solution for different numbers of iterations. A number of tests were also performed by perturbing the functional unit and the system was found to be stable.

The reason this system is stable despite being ill-conditioned is probably related to the fact that it is very sparse (1.4% of elements are non-zero) and consequently the columns are nearly linearly-dependent (causing the matrix to appear singular). It was also found that the data directly from ecoinvent, which forms the foundation for the data used in this article, is ill-conditioned. Consequently, for these systems it is best to verify that the solution is accurate by back-substitution rather than relying on the condition-number; this approach is also computationally faster.

3 Monte-Carlo Analysis

Monte-Carlo analysis is a technique that propagates known parameter uncertainties through a calculation to give an uncertainty distribution on the output variables (Morgan and Henrion 1990). Consequently, Monte-Carlo analysis is an ideal method for quantifying parameter uncertainty in LCA studies (e.g. Huijbregts et al. 2001, 2003) and IOA studies (e.g. Bullard and Sebald 1988, Lenzen 2001). However, it seems Monte-Carlo analysis is rarely performed in LCA and IOA studies; one possible reason for this is a belief that Monte-Carlo analysis is computationally slow (c.f. Heijungs et al. 2005). In this section, I apply Algorithm 3 to reduce the computational times of conventional Monte-Carlo studies. Ciroth et al. (2004) have a similar motivation, but do not use the matrix approach which makes there method not as viable for hybrid LCA systems.

The algorithms presented in the previous section are particularly relevant in performing a Monte-Carlo analysis. The accuracy of a Monte-Carlo simulation generally increases with the square root of the number of samples taken; a reliable Monte-Carlo analysis might consist of 10,000 or more samples[12]. The algorithms presented above give the opportunity for large increases in the sample size in a Monte-Carlo simulation.

I applied a simple Monte-Carlo analysis as described in Algorithm 3. The algorithm has several functions which need explaining. The variable N is the number of samples in the Monte-Carlo calculation. I have assumed that F, A, and y are stored as sparse matrices. For performing loops on a sparse matrix it is considerably faster to loop through the array representation of a sparse matrix than the full matrix (e.g. Saad 1996). The function FIND() is an operation that returns an array representation of a sparse matrix[13] (Saad 1996); that is, the row and column indices and the value of the non-zero entries are returned. The function SPARSE() converts the array representation of a matrix back to the sparse matrix representation[14]. In the procedure RANDOM(), the function PDF(V,σ) returns randomly distributed numbers with a given probability distribution; in the example here, a normal distribution. The error is taken as three standard deviations which covers 99% of the distribution.

[12] The sample size will vary for different problems and it is difficult to make a generalization of how many samples are adequate. Heijungs et al. (2005) suggest 500 to 1,000 samples. This may not always be an adequate number of samples. Huijbregts et al. (2003) use 10,000 samples.

[13] As in the Matlab find routine.

[14] As in the Matlab sparse routine.

Algorithm 3: MONTE_CARLO(F, A, y, N, *error*, *tolerance*, *max_iterations*)

```
procedure RANDOM(R, C, V, σ)
comment: This procedure gives a probability
distribution to the non-zero elements
for e ← 1 to LENGTH(R)
  V_new[e] ← PDF(V[e], σ[e])
end
return SPARSE(R, C, V_new)
main
{A_rows, A_columns, A_values} ← FIND(A)
{V_rows, V_columns, V_values} ← FIND(V)
A_σ ← error/3 A_values
F_σ ← error/3 F_values
for i ← 1 to N
  A_sample ← RANDOM(A_rows, A_columns, A_values, A_σ)
  F_sample ← RANDOM(F_rows, F_columns, F_values, F_σ)
  E[i] ← CALCULATE_IMPACT(F_sample, A_sample, y, tolerance,
  max_iterations)
end
return E
```

For the Monte-Carlo analysis I only apply distributions to the non-zero elements. The easiest way to operationalize this process is to convert the matrices into a sparse representation. This is performed in the first two lines of Algorithm 3; for each non-zero element in A, A_{rows} contains the row index, $A_{columns}$ contains the column index, and A_{values} the value of the non-zero element. For the demonstrative purpose of this example, each element in the matrices is given a constant percentage error with a normal distribution[15]. The standard deviation of the coefficients is given by F_σ and A_σ; the *error* is assumed to be three standard deviations. Once the data has been initialized, the algorithm then calculates the emissions for each sample of the Monte-Carlo. The procedure RANDOM produces a new sample distribution of the matrices and then CALCULATE_IMPACT is used to determine the environmental impacts with a given accuracy (*tolerance*) for each sample. The algorithm returns a vector containing the environmental impacts for each Monte-Carlo sample.

Algorithm 3 was demonstrated on the system described above and the results are shown in **Table 3**. I used a normally distributed error of 25%, with the error defined as three standard deviations. The Monte-Carlo results stabilize at around 1,000 samples, suggesting that 1,000 samples are adequate for this system. This calculation takes approximately 90 seconds for the 3225×3225 system described above. Although, the time penalty for using more samples is not great. With smaller systems it is likely that Monte-Carlo simulations for 10,000 or more samples will take only a matter of minutes.

The computationally slowest part of the algorithm is in the calculation of the randomly distributed matrices, that is, the function RANDOM. There are many options for speeding up this function. For instance, a smaller sample size can be used by dividing the sample space into equiprobable intervals; such as in Latin Hypercube Sampling[16] (Morgan and Henrion 1990). Sampling techniques give a more uniform distribution for a given sample size allowing the sample size to be reduced; however, they are also computationally expensive. Thus, sampling techniques are only used if the computation times with a reduced sample size offset the time to do the sampling. For our example problem, the procedure CALCULATE_IMPACT is sufficiently fast so that it is slower to reduce the sample size using sampling techniques.

4 Conclusion

In this article I have demonstrated the use of iterative algorithms for solving LCA and IOA systems. The algorithms were demonstrated on a sparse 3225×3225 hybrid LCA system and they performed several orders of magnitude faster than direct methods of solution. Several issues relating to the convergence of the algorithm were discussed and it was shown that the structure of LCA and IOA systems will generally lead to convergence. The algorithm was then used to perform a Monte-Carlo simulation on the same system. It was demonstrated how efficient methods of solving the LCA system can be used to employ a large number of samples in a Monte-Carlo analysis. The algorithms presented here make it possible to perform Monte-Carlo analysis on most LCA and IOA systems in a matter of seconds to minutes.

Acknowledgements. Anders Hammer Strømman provided the data for the simulations performed in the article. Two reviewers provided useful comments which greatly improved the article.

[15] A log-normal (or other) distribution may be more appropriate in many practical studies(e.g. Huijbregts et al. 2003; Bullard and Sebald 1988).

[16] The Matlab routine lhsnorm uses Latin Hypercube Sampling.

Table 3: The computational times and accuracies for the Monte-Carlo simulation. The mean values estimate $F(I\text{-}A)^{-1}y$, Std is the standard deviation, and the percentage error is given by three standard deviations

Samples	Mean	Std	Error (%)	Time
exact	**103.5**			
10	102.4	7.1	20.8	00:00:01
100	104.2	9.2	26.4	00:00:10
1,000	103.3	8.4	24.4	00:01:30
10,000	103.5	8.7	25.1	00:15:00
100,000	103.5	8.7	25.1	02:30:00

Appendix A: Normalized systems of linear equations for LCA

In this article I have used a normalized system of linear equations to represent an LCA system. Some authors prefer to use an un-normalized system. In this appendix I briefly describe the differences between the two systems.

Consider an LCA system written in an un-normalized form (Heijungs and Suh 2002)

$$\tilde{A}s = y \tag{11}$$

where y is the demand on the LCA system, s is a scaling vector for the different processes, and $\tilde{A}$ is the un-normalized 'technology' of the LCA system. I have assumed that each process in the un-normalized system has only one output through allocation (Heijungs and Suh 2002) and the processes are ordered to have the output on the diagonal of $\tilde{A}$.

The normalized system, (1), can be obtained by using

$$A = \left(\tilde{D} - \tilde{A}\right)\tilde{D}^{-1} = I - \tilde{A}\tilde{D}^{-1} \tag{12}$$

where $\tilde{D}$ is a matrix with the diagonal elements of A on the diagonal and hence the scaling vector s is replaced by the output,

$$x = \tilde{D}s \tag{13}$$

Further, the emission intensity also needs to be normalized, compare with (3), so that

$$F = \tilde{F}\tilde{D}^{-1} \tag{14}$$

It is worth elaborating on a few differences in the normalized and un-normalized systems. In the un-normalized system, (11), the output of the processes are placed on the diagonal of $\tilde{A}$ and the off-diagonal terms are negative to represent 'inputs' into the production processes. The elements of s represent the 'share' of each process that is needed to produce the functional unit, y. In the normalized system, the technology matrix is normalized so that each process gives one-unit of output. This allows the outputs to be removed from matrix and the inputs represented as positive entries (12). Each column of the normalized matrix represents the inputs to produce one-unit of output of each process. Therefore the x represents the total output of each process. Any analysis should produce the same environmental impacts, but the outputs differ as in (13).

The use of A and $\tilde{A}$ by different authors probably results from different backgrounds. The matrix approach to LCA was originally presented in the un-normalized form (Heijungs 1994, Heijungs and Suh 2002). IOA data is often compiled in an un-normalized form, but analysis is performed using normalized data (cf. Leontief 1970, Miller and Blair 1985, United Nations 1999). An advantage of using the normalized form is that it allows direct application of many methods already established in IOA.

References

Baumann H, Tillman A-M (2004): The hitch hiker's guide to LCA: An orientation in life cycle assessment methodology and application. Studentlitteratur, Lund

Berger W, Saibel E (1957): Power series inversion of the Leontief matrix. Econometrica 25 (1) 154–165

Bullard CW, Sebald AV (1988): Monte Carlo sensitivity analysis of input-output models. The Review of Economics and Statistics 70 (4) 708–712

Bureau of Economic Analysis (2005): Bureau of Economic Analysis: Industry economic accounts. Online database: Accessed 13 January, 2005, Bureau of Economic Analysis, URL <http://www.bea.doc.gov/bea/dn2/i-o.htm>

Ciroth A, Fleischer G, Steinbach J (2004): Uncertainty calculation in life cycle assessments: A combined model of simulation and approximation. Int J LCA 9 (4) 216–226

Datta B (1995): Numerical linear algebra and applications. Brooks/Cole Publishing Company

Frischknecht R, Jungblush N (2004): Overview and methodology. ecoinvent report No. 1, Swiss Centre for Life Cycle Inventories, URL <http://www.ecoinvent.ch/>

Hackbusch W (1994): Iterative Solution of Large Sparse Systems of Equations. Springer-Verlag, New York

Hawkins D, Simon HA (1949): Note: Some conditions of macroeconomic stability. Econometrica 17 (3) 245–248

Heijungs R (1994): A generic method for the identification of options for cleaner production. Ecological Economics 10, 69–81

Heijungs R, Suh S (2002): Computational structure of life cycle assessment. Kluwer Academic Publications, Dordrecht, The Netherlands

Heijungs R, Suh S, Kleijn R (2005): Numerical approaches to life cycle implementation: The case of the ecoinvent'96 database. Int J LCA 10 (2) 103–112

Hendrickson C, Horvath A, Joshi S, Lave L (1998): Economic input-output models for environmental life-cycle assessment. Environmental Science and Technology 32 (7) 184A–191A

Hertwich EG (2005): Lifecycle approaches to sustainable consumption: A critical review. Environmental Science and Technology 39 (13) 4673–4684

Hocking MB (1991): Paper versus polystyrene: A complex choice. Science 251, 504–505

Huijbregts MA, Gilijamse W, Ragas AM, Reijnders L (2003): Evaluating uncertainty in environmental life-cycle assessment. A case study comparing two insulation options for a Dutch one-family dwelling. Environmental Science and Technology 37, 2600–2608

Huijbregts MA, Norris G, Bretz R, Ciroth A, Maurice B, von Bahr B, Weidema B, de Beaufort AS (2001): Framework for modelling data uncertainty in life cycle inventories. Int J LCA 6 (3) 127–132

Joshi S (1999): Product environmental life-cycle assessment using input-output techniques. Journal of Industrial Ecology 3 (2–3) 95–120

Lave L, Cobas-Flores E, Hendrickson C, McMichael F (1995): Using input-output analysis to estimate economy-wide discharges. Environmental Science and Technology A 29 (9) 420A–426A

Lenzen M (2001): A generalized input-output multiplier calculus for Australia. Economic Systems Research 13 (1) 65–92
Lenzen M (2003): Environmentally important paths, linkages and key sectors in the Australian economy. Structural Change and Economic Dynamics 10 (6) 545–572
Leontief W (1970): Environmental repercussions and the economic structure: An input-output approach. The Review of Economics and Statistics 52 (3) 262–271
Miller R, Blair P (1985): Input-output analysis: Foundations and extensions. Englewood Cliffs, NJ, Prentice-Hall
Morgan M, Henrion M (1990): Uncertainty: A guide to dealing with uncertainty in quantitative risk and policy analysis. Cambridge University Press
Nakamura S, Kondo Y (2002): Input-output analysis of waste management. Journal of Industrial Ecology 6 (1) 39–63
Nicholson WK (1990): Elementary linear algebra with applications. PWS-Kent Publishing Company
O'Neill MJ, Wood RJ (1999): An alternative proof of the Hawkins-Simon Condition. Asia-Pacific Journal of Operational Research 16, 173–183
Peters GP, Hertwich EG (2006): A comment on 'Functions, commodities and environmental impacts in an ecological-economic model'. Ecological Economics, Forthcoming
Saad Y (1996): Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston
Seneta E (1973): Non-negative matrices: An introduction to theory and applications. George Allen & Unwin Ltd
Stewart GW (1973): Introduction to matrix computations. Academic Press
Strømman AH, Solli C, Hertwich EG (2006): Hybrid life-cycle assessment of natural gas based fuel chains for transportation. Environmental Science and Technology 40 (8) 2797–2804
Suh S (2004): Functions, commodities and environmental impacts in an ecological-economic model. Ecological Economics 40 (4) 451–467
Suh S (2006): Reply: Downstream cut-offs in Integrated Hybrid Life Cycle Assessment. Ecological Economics (forthcoming)
Suh S, Lenzen M, Treloar GJ, Hondo H, Horvath A, Huppes G, Jolliet O, Klann U, Krewitt W, Moriguchi Y, Munksgaard J, Norris G (2004): System boundary selection in life-cycle inventories using hybrid approaches. Environmental Science and Technology 38 (3) 657–664
United Nations (1993): System of National Accounts 1993. United Nations
United Nations (1999): Handbook of input-output table compilation and analysis. Studies in Methods Series F, No 74, Handbook of National Accounting, United Nations
Varga RS (2000): Matrix iterative analysis, 2nd Edition. Springer
Watkins DS (2002): Fundamentals of Matrix Computations, 2nd Edition. Wiley-Interscience, New York
Waugh FV (1950): Inversion of the Leontief matrix by power series. Econometrica 18 (2) 142–154
Wood RJ, O'Neill M (2002): Using the spectral radius to determine whether a Leontief system has a unique positive solution. Asia-Pacific Journal of Operational Research 19, 233–247
Wood RJ, O'Neill MJ (2004): An always convergent method of finding the spectral radius of an irreducible non-negative matrix. Australian & New Zealand Industrial and Applied Mathematics Journal 45, C475–C485

Received: May 4th, 2005
Accepted: May 29th, 2006
OnlineFirst: June 26th, 2006

Int J LCA 10 (2) 103 –112 (2005)

Numerical Approaches to Life Cycle Interpretation

The case of the ecoinvent'96 database

Reinout Heijungs*, Sangwon Suh and René Kleijn

Institute of Environmental Sciences, Leiden University, P.O. Box 9518, 2300 RA Leiden, The Netherlands

* **Corresponding author** (heijungs@cml.leidenuniv.nl)

DOI: http://dx.doi.org/10.1065/lca2004.06.161

Goal, Scope and Background. To strengthen the evaluative power of LCA, life cycle interpretation should be further developed. A previous contribution (Heijungs & Kleijn 2001) elaborated five examples of concrete methods within the subset of numerical approaches towards interpretation. These methods were: contribution analysis, perturbation analysis, uncertainty analysis, comparative analysis, and discernibility analysis. Developments in software have enabled the possibility to apply the five example methods to explore the much-used ecoinvent'96 database.

Discussion of Methods. The numerical approaches implemented in this study include contribution analysis, perturbation analysis, uncertainty analysis, comparative analysis, discernibility analysis and the newly developed key issue analysis. The data used comes from a very large process database: ecoinvent'96, containing 1163 processes, 1181 economic flows and 571 environmental flows.

Conclusions. Results are twofold: they serve as a benchmark to the usefulness and feasibility of these numerical approaches, and they shed light on the question of stability and structure in an often-used large system of interconnected processes. Most of the approaches perform quite well: computation time on a moderate PC is between a few seconds and a few minutes. Only Monte Carlo analyses may require much longer, but even then it appears that most questions can be answered within a few hours. Moreover, analytical expressions for error propagation are much faster than Monte Carlo analyses, while providing almost identical results. Despite the fact that many processes are connected to each other, leading to the possibility of a very unstable system and very sensitive coefficients, the overall results show that most results are not extremely uncertain. There are, however, some exceptions to this positive message.

Keywords: Contribution analysis; discernibility analysis; ecoinvent'96; key issue analysis; life cycle interpretation; perturbation analysis; sensitivity analysis; uncertainty analysis